A Subsystem that Behaves as an LXI Instrument

Brandon Thorpe
Military and Aerospace Division
Teradyne, Inc
North Reading, MA
brandon.thorpe@teradyne.com

This paper describes an intelligent subsystem that integrates into a larger test system as if it were a single complex instrument. Actually, the subsystem contains a controlling embedded PC and various instruments that conform to the PXIe test standard working in tight coordination to perform complex tasks. The test station computer communicates with the subsystem with a high-level IVI driver via Ethernet.

LXI is an attractive choice for integrating test instruments into a larger test system largely due to the interconnection simplicity provided by Ethernet. LXI is typically used with individual instruments, not a group of instruments controlled with an embedded PC. While it is possible to control all of the PXIe instruments directly from the main test station computer, the task of integrating the functions would need to be repeated for each test system design. A novel solution is to configure the subsystem as a single LXI compliant device with extensive localized and reusable capabilities.

The PXIe chassis that houses the subsystem instruments is controlled with an embedded PC, which hosts an LXI discovery service and IVI remote driver. The services are implemented using C, C# and .NET on an embedded 64-bit Windows 7 operating system. By developing LXI code that runs on the embedded PC housed within the PXIe chassis, we treat the entire subsystem as an LXI compliant device.

This paper shall discuss the high level API with which the subsystem is controlled, as well as the architecture of the services running on the embedded controller that make the subsystem LXI compliant.

I. Introduction

LXI, or *LAN Extensions for Instrumentation*, is a set of extensions that provide increased interoperability and functionality of LAN enabled test and measurement equipment. The LXI Consortium was founded in 2004, and the LXI 1.0 specification was released in September 2005. Instruments that are LXI compliant can be discovered, configured, and controlled over Ethernet as opposed to GPIB, VXI, or PXI. LXI is suitable for customers that don't like specialized cards and cables. LXI is also suitable for customers who don't like expensive slot 0 controllers. The LXI consortium was formed after instruments with USB and LAN interfaces became enormously popular. Without a triggered backplane, LXI devices are triggered much like traditional GPIB instruments, where users can send a trigger over the Trigger Bus to start a measurement from an external event.

II. High Speed Subsystem

Figure 1: High Speed Subsystem device. To the left end of the chassis is the embedded controller, in the slots are Teradyne LVDS core instruments.

978-1-4244-9363-0/11/$26.00 ©2011 IEEE

The High Speed Subsystem is a subsystem which is controlled by an embedded 64-bit Windows 7 computer. The computer acts as the slot 1 controller for a PXIe chassis, housing multiple sub-instruments. Each High Speed Subsystem has at least one LVDS core sub-instrument and may have multiple expansion instruments with unique test capabilities. Data acquired from a unit under test can be processed using either the programmable microprocessor on the core instrument, or the embedded controller in the chassis. Our goal was to develop an interface such that this entire enclosure could be seen by a Test Station PC as a single LXI compliant device. Since the embedded controller of the subsystem is a relatively high powered PC running Windows 7, the technologies at our disposal for conforming to LXI compliance, and our final API are in many ways unique when compared to traditional LXI devices.

III. ADVANTAGES

The conventional use of a PXIe chassis equipped with an embedded slot 1 controller is to connect a monitor and other peripherals in order to program subsystem instruments locally. In a large scale test configuration, where many instruments are controlled from a *Tester PC*, it is not realistic to host or control the auxiliary instrument drivers on the embedded controller. By creating an LXI interface for the PXIe chassis, it is possible to treat the entire subsystem as any other device in the test configuration controlled by the main *Tester PC*. Also, it is possible to expose virtually all of the embedded controller's capabilities through a programmatic interface, which is paramount for test developers interfacing with other devices such as power supplies and auxiliary instrumentation.

IV. COMPLIANCE

The High Speed Subsystem targets class C compliance within the LXI specification revision 1.3 [7], which has major disparities with LXI 1.4. LXI 1.4 has inherent support for IPv6 as well as the notion of extended versus core features as opposed to a class based hierarchy.

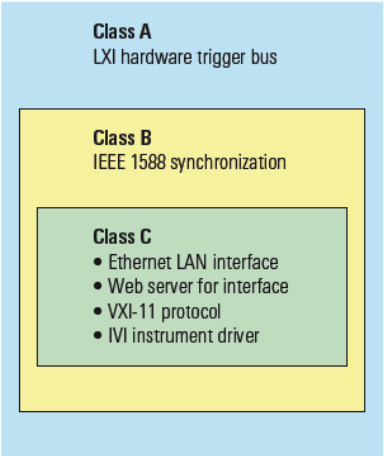

Figure 3: LXI 1.3 class based compliance hierarchy.

Due to the nature of the High Speed Subsystem, Class C is befitting. LXI is used for communication between the Test Station PC and the *High Speed Subsystem* device, this communication does not involve triggering or high bandwidth data transfer.

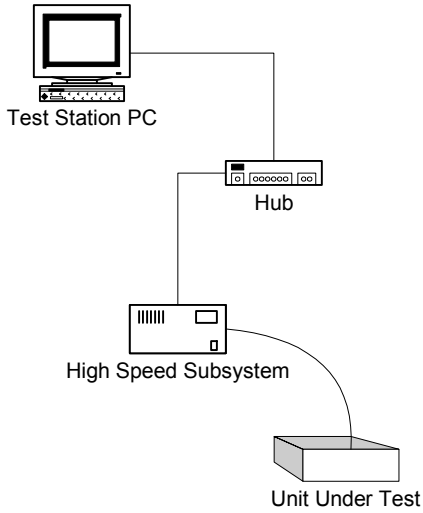

Figure 1: This figure illustrates a simple network involving a test station controlling an HSS device with LXI.

All triggering and high bandwidth data acquisition tasks are isolated to the region of communication between the *High Speed Subsystem*, its PXIe based sub-instruments, and the units under test. Data transfer between subsystem instruments and the embedded controller uses PXIe, which is an extremely fast and precise interface. The Test Station's role is merely to assert high level control of the *High Speed Subsystem*. Class C compliance requires the device to support VXI-11 discovery, host an advanced web interface, and be programmable with an IVI compliant driver. LXI does not require any functionality in the driver beyond what IVI requires. That being said, it is possible to have an LXI compliant device that provides little to no real remote programming capabilities. Our IVI-C driver required additional capabilities beyond those specified by IVI

in order to access the devices file system and enumerate subsystem instrumentation.

V. LXI DISCOVERY

LXI compliant devices must be discoverable by a client using the VXI-11 protocol. In theory, a user can open NI-MAX or any LXI discovery tool and search for LAN connected or TCP/IP instruments. If an instrument is LXI compliant, and on the same subnet as the client machine, it will appear on the list of available instruments.

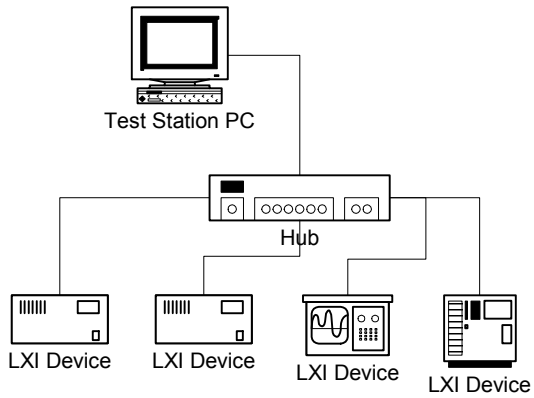

Figure 2: The figure above shows multiple LXI discoverable devices residing on the same LAN as a test station PC.

From that point, a user can add the instrument as a resource and access it using VISA in their application. The capabilities exposed through VISA may be very limited since LXI requires that only VXI-11 identification be supported. In the case of the *High Speed Subsystem*, the VXI-11 protocol was not an ideal solution for many of the remote capabilities that the device would expose. The alternative is discussed in the IVI Compliant Driver section of this paper. The *High Speed Subsystem* does support VXI-11 discovery for LXI compliance.

VXI-11 was originally designed to mimic GPIB. It was added to VISA specification in 2000 for LAN connected instrumentation support. VXI-11 allows ASCII messages, including IEEE 488.2 messages, and IEEE 488.1 instrument control messages to be passed between a controller and a device over a TCP/IP network built on ONC-RPC (*Open Network Computing Remote Procedure Calls*).

ONC-RPC is a technology traditionally seen on Unix systems, though, there are Microsoft implementations available. With ONC-RPC, a developer defines the client/server procedure call interface in a definition (*.x) file, and then a generation tool will create code stubs for the server implementation which will be hosted on the LXI device.

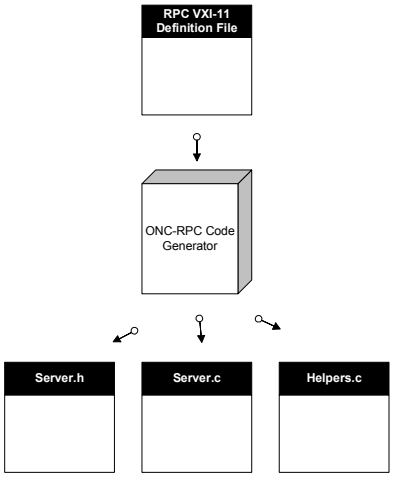

Figure 3: The RPC generator tool converts an interface definition *.x file into a code base shell for a simple VXI-11 server. This simple server is used only for LXI discovery in the *High Speed Subsystem*.

The VXI-11 specific interface is defined in the VXI-11 specification and can be copied directly into a definition file for the RPC generator tool.

In the *High Speed Subsystem LXI Discovery Service*, **create_link, destroy_link, device_lock, device_unlock, device_read, and device_write** are the only RPC procedures supported and implemented. Link maitnence is required to handle multiple requests for identification simultaneously. Read and write are required for identification query responses. After implementing the code stubs and deploying the server on the device, clients can then perform reads and writes on the server through client side VISA implementations that support VXI-11 such as NI-VISA and Agilent VISA.

For LXI Class C compliance, only one remote command need be supported. This is the identify command, which is accomplished by performing an RPC device write of "*IDN?" An LXI compliant device must reply with it's identity upon such a write. The implementation of these functions has some important rules which can be found in VXI-11 specifications [1] [2] [3].

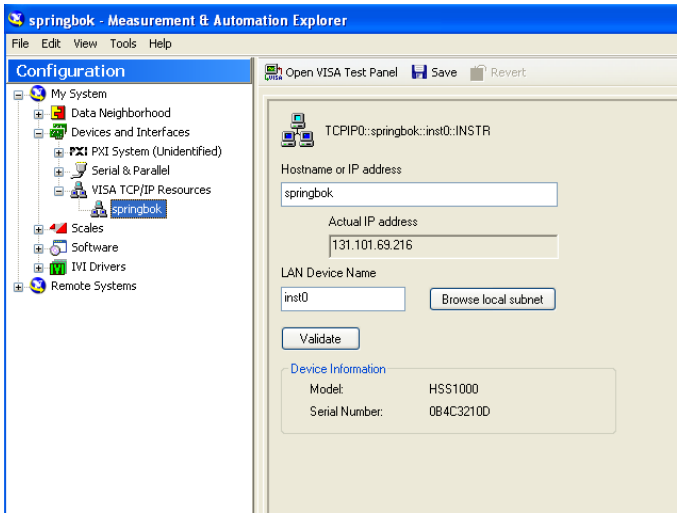

Figure 6: HSS device, hostname "springbok", discovered on the LAN with NI-MAX.

In the case of the *High Speed Subsystem*, this LXI discovery server is built as a Windows service application that begins running at device boot. Effectively this makes the device discoverable as long as it is on and the operating system has started. The service application queries the instrument model, software, and serial number from the Windows registry, and reports that information to clients when queried for identification.

VI. IVI COMPLIANT DRIVER

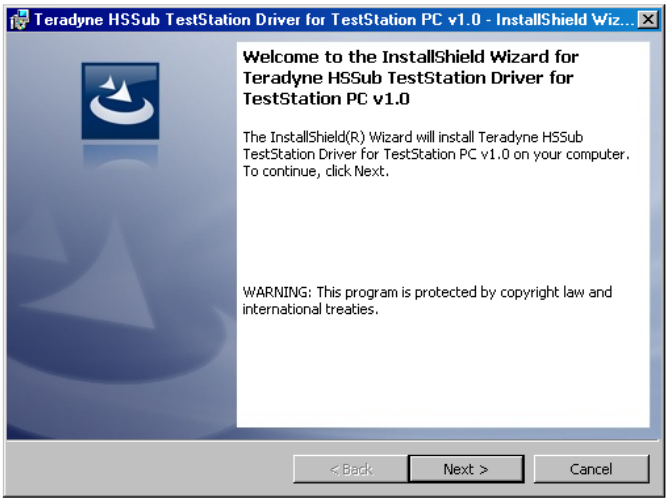

Figure 7: Windows installer for the Teradyne *High Speed Subsystem* LXI (IVI-C) driver installation.

An LXI compliant device must have an IVI-C or IVI-COM compliant driver. *High Speed Subsystem* supports an IVI-C compliant driver coined the *Test Station Driver* due to the fact that it is designed to be installed and run on the Test Station PC. An IVI-C driver must export functions and attributes that comply with *IVI-3.2 Inherent Capabilities Specification* [4] such as Session Init and Close functions, Attribute Set and Get functions, Lock/Unlock, Self Test/Reset and Error handling functions. In addition, an IVI-C driver can export functions and attributes that are specific to the instrument. Again, there are no LXI driver requirements beyond those stated in the IVI Inherent Capabilities specification. If you can create a basic IVI driver for your device, then you have done enough to create an LXI compliant IVI driver. However, this bare functionality is not sufficient for the *High Speed Subsystem*. The Test Station Driver API exposes functionality for remote file system control, remote application execution, and subsystem instrument enumeration. The implementation of this functionality was considerably simplified by the fact that our LXI *device* is a subsystem controlled by an embedded Windows based PC.

The most interesting aspect of the *Test Station Driver* API is remote application execution. Actual test code which uses the drivers of the subsystem instrumentation is transmitted from the *Tester PC* to the embedded controller and executed on the embedded controller. All core instrument use, data processing, and in effect all "testing" done by the *High Speed Subsystem* is done between the embedded controller and the PXIe instrumentation. Test results are transmitted back to the *Tester PC* after the heavy duty operations are completed. This is important because it highlights the fact that LXI based triggering is not a requirement for the *High Speed Subsystem*. All triggering is done between code executed on the embedded controller and the subsystem instrumentation via the PXIe backplane.

To expose these capabilities of the *High Speed Subsystem* device remotely to a Test Station over Ethernet, the driver uses Microsoft's .NET Remoting technology. In some cases, ONC-RPC, the same technology used for VXI-11 discovery, is used to accomplish these remote driver operations on LXI devices. However, since the *High Speed Subsystem* is running Windows, .NET Remoting was a somewhat simpler and more elegant solution. The *High Speed Subsystem Test Station Driver* is an IVI-C native library. All of the .NET Remoting functionality is packaged into a managed C++ library which is invoked by the native driver.

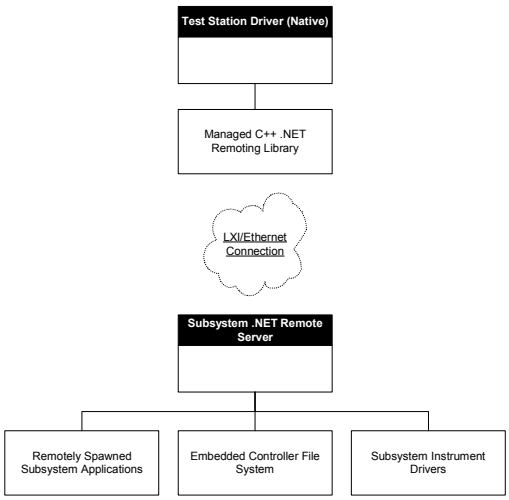

Figure 8: Subsystem instrument enumeration and embedded PC capabilities exposed remotely to Test Station PC via Subsystem Server and Test Station Driver.

On the device side, a .NET Remoting server is running alongside the LXI discovery service. This server, which also begins running during device boot, provides all the remote capabilities accessed by the Test Station driver.

VII. WEB INTERFACE

An LXI compliant device must host a web-server with W3C compliant web pages that embody an interface for configuring the device on the LAN. There are several broad requirements as well as some detailed requirements that can be found in the LXI specification [7]. The general requirements are listed below:

- Web pages shall be W3C compliant
- Protocol shall be HTTP
- Connections shall be accepted on port 80
- Mandatory welcome page display items
- LAN Configuration web page
- Security mechanism
- Visible LXI logo

The *High Speed Subsystem* uses the Microsoft Windows IIS (Internet Information Services) web server, and hosts active ASP.NET pages to provide the required functionality. ASP.NET is Microsoft's .NET version of Active Server Pages, which allows C# code to run behind dynamic HTML pages on the server. This enables the *High Speed Subsystem* to have a sophisticated web interface without many of the limitations normally associated with hosting a web server on an instrument.

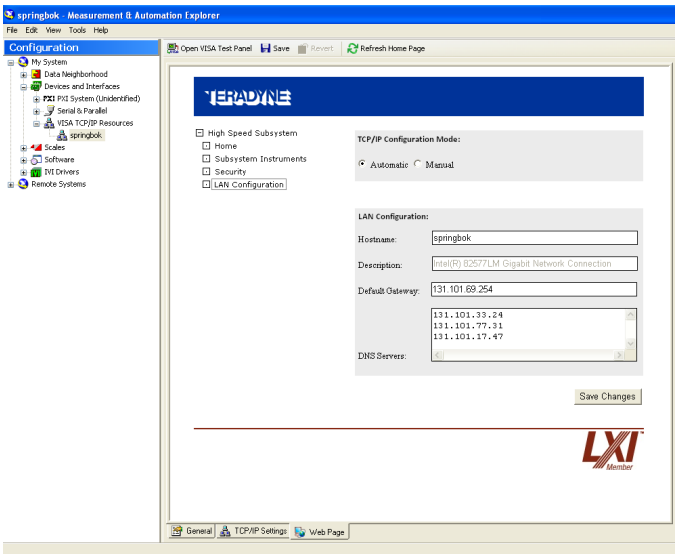

Figure 9: HSS device web interface as seen through NI-MAX.

The web interface must display general instrument information on the welcome page. This information includes the instrument model, manufacturer, serial number, description, etc. In the case of the *High Speed Subsystem*, this information is stored in the Windows registry on the device. Enumeration of subsystem instruments within the PXIe chassis is accomplished by performing Teradyne driver calls and 3rd party instrument driver calls at run time with ASP.NET.

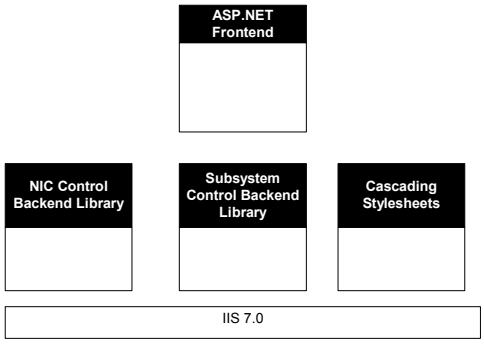

Figure 10: Subsystem instrument enumeration and device NIC properties exposed via ASP.NET web interface.

The device's web interface must also provide access to the NIC (Network Interface Card) in order to configure the LAN settings. The LAN settings that must be configurable are listed below:

- Hostname
- Device Description on NIC
- TCP/IP Configuration Mode
- IP Addresses
- Subnet Mask
- Default Gateway
- DNS Servers

The *High Speed Subsystem* web server contains a backend library that uses the *Win32_NetworkAdapterConfiguration* management class. The web pages interface with the library using ASP.NET C# code behind the HTML. The *Win32_NetworkAdapterConfiguration* management class provides all the functionality required for LXI network configuration on a Windows based device.

An LXI Class C compliant device's web interface must also provide a security mechanism so that NIC settings can't be maliciously tampered with. This is accomplished by using Windows Authentication functions to allow ASP.NET code to run as an impersonated user once a password was provided through the web interface.

VIII. Conclusion

In conclusion, providing an LXI interface to a subsystem is convenient to the end user in that it gives them a standardized way to locate, configure and program the subsystem. By using a high powered Windows based embedded PC as the chassis slot 1 controller, features were provided in addition to the standard LXI requirements that would otherwise be exceedingly difficult to implement.

References

[1] VMEbus Extensions for Instrumentation: TCP/IP-VXIbus Interface Specification, VXI-11.1, Revision 1.0.

[2] VMEbus Extensions for Instrumentation: TCP/IP-IEEE 488.1 Interface Specification, VXI-11.2, Revision 1.0.

[3] VMEbus Extensions for Instrumentation: TCP/IP-IEEE 488.1 Interface Specification, VXI-11.3, Revision 1.0.

[4] IVI Interchangeable Virtual Instruments: IVI-3.2 Inherent Capabilities Specification, June 9, 2010 Edition, Revision 2.0.

[5] NI-VISA Programmer Reference Manual, March 2003 Edition.

[6] ICS Electronics: VXI-11 Tutorial and RPC Programming Guide.

[7] LAN Extenstions for Instrumentation: LXI Standard, October 30, 2008 Edition, Revision 1.3.

[8] Advanced .NET Remoting 2cnd Edition, Ingo Rammer and Mario Szpuszta, 2005.